In the Specification:

The paragraph on page 4, spanning lines 4-11, has been amended as shown below:

The module is the basic functional unit and represents a container of behavior that may be implemented by one or more computers running one or more software programs. For instance, in the context of a Website, one module might represent a front end that renders HTML pages, another module might represent a login database, and another module might represent a mailbox program. A port is a service access point for the module. All communications into and out of the module go ~~goes~~ through a port. A wire is the logical binding that defines an allowed communication route between two ports.

The paragraph on page 4, spanning lines 12-23, has been amended as shown below:

While the model consists of the three basic components described above (namely modules, ports, and wires), the model can be augmented with numerous extensions, specializations of the basic components. For example, a store is a basic unit of storage and a specialization of the module. A store represents a logical amount of storage, which may be implemented by any number of physical disks or other storage media. Like the module, the store represents behavior, in this ~~the~~ case the ability to save and retrieve data. Also like the module, the store can communicate with other modules and stores through ports and wires. A store differs from a module in that it is labeled with additional attributes such as the amount of storage required, required access speed, or a minimum number of outstanding storage requests. The store extends the model by adding a specialized type of module with additional semantic information.

The paragraph on page 9, spanning lines 15-23, has been amended as shown below:

Each module 202 defines a unit of scaling. While one module logically represents a functional operation of the Service, the module may be deployed to any number of computers when actually implemented. In this way, the module is scale-independent, allowing the number of underlying computers used to implement the module to change ~~at~~ over time. When converted to a physical implementation, "module instances" are created from the modules. The module instances are assigned a unique identifier and maintain ancestral data regarding which module created them. The module instances of simple modules are often called "engines", which are software programs that run on an individual computer.

The paragraph spanning page 13, line 22, through page 14, line 2, has been amended as shown below:

The store 316 has a port 340 to communicate with the primary 312 and secondary 314 SQL modules ~~312~~ and an event sink 342 to receive event messages from the fail-over policy module 310. A wire 344 interconnects the store port 340 with the ports 322 and 332 of the primary and secondary SQL modules, respectively.

The paragraph spanning page 19, line 24, through page 20, line 9, has been amended as shown below:

In one embodiment, the developer writes management policy, which issues commands on the schema 518 to create new instances of modules, port, and wires to deploy the Internet Service. Developers may choose to write management policy instead of using fully automatic logical-to-physical converter 520 when they want finer control over the growth and management of the Internet Service. The management code issues commands using the namespace defined by the schema 518, but the commands operate on individual module, port, and wire instances. The commands are still dispatched through the converter 520, which allocates nodes to the management policy. Whether operating automatically or driven by management policy code, the ~~convert~~ converter 520 records in the instance-tracking database 522 the individual instances of modules, port, and wires.

The paragraph on page 22, spanning lines 10-19, has been amended as shown below:

Once a logical model is created, an automatic computer-based deployment system uses the logical model to deploy various computer/software resources to implement the Internet Service. The deployment system converts each of the model components into one or more instances that correspond to physical resources, such as nodes of a distributed computer system that are loaded with specific types of software to implement the function represented by the model components. The deployment system initially installs an Internet Service on the physical resources according to the logical model. It then dynamically and automatically modifies the resources used to implement the Internet Service ~~in~~ on an ongoing basis as the operating parameters of the application change.

The paragraph on page 23, spanning lines 15-23, has been amended as shown below:

The management policy 702 implements one or more policies devised by the developer or operator of the Internet Service. The policies specify when instances derived from the logical model should be created, manipulated, and destroyed. The management policy monitors various events generated by the nodes and implements policy decisions regarding how to handle the events. By specifying when and ~~what~~ which instances should be created (or destroyed), the management policy 702 effectively dictates when hardware/software resources 706 should be added (or removed) to support the changing demands of the Internet Service.

The paragraph spanning page 23, line 24, through page 24, line 14, has been amended as shown below:

The core converter 704 implements the policy decisions made by the management policy 702. The runtime converter 704 has a service running state 710 that tracks all instances of the model components currently in existence. That is, the service running state 710 tracks the elements in the physical world with respect to the logical model. The service running state 710 maintains a copy of the logical model, such as online retailing model 400 of Fig. 4. The logical model is created by the modeling system described above with respect to Figs. 5 and 6. The current instances are maintained in the instance-tracking database 522. The records in instance-tracking database 522 include such information as ~~identify~~ identity of the instance, the name of the logical component from which it is derived, the node on which it is running, the network addresses representing the ports of the modules and module extensions, such as stores, type of software loaded on the node, various protocols supported by the instance, and so forth. The instance-tracking database 522 tracks not only module instances, but also port ~~instance~~ instances, wire instances, and can also track instances of extensions such as stores, event ports, and event wires.

The paragraph on page 24, spanning lines 18-25, has been amended as shown below:

The core logical-to-physical converter 704~~,~~ manages the creation of new instances in the physical world from the model components specified in the logical model 400 through a loader 722. A loader 722 carries out the configuration of newly allocated resources to the functions dictated by the new instances. In this manner, when another instance of a component is desired, the management policy 702 ~~720~~ communicates with the resource manager 716 to allocate a node from the resource pool and with the loader 722 to load the appropriate software programs onto the node.

The paragraph on page 26, spanning lines 5-9, has been amended as shown below:

Fig. 8 shows an example of a portion of the logical model 400 of Fig. 4 being converted into actual instances. To illustrate the conversion, the front end module 402 and the order processing module 406 are extracted from the online retailer service application 400 (Fig. 4). A wire 442 interconnects the two modules by logically coupling the ports 422 and 440.

The paragraph spanning page 26, line 20, through page 27, line 2, has been amended as shown below:

The ports 422 and 440 represent network ports with protocols and roles within protocols in the logical world. They translate to the physical world as a set of network addresses used to communicate with the software at the respective modules. For instance, the physical translation of a logical port 422 might be IP Port 804(n) ~~80~~, using HTTP (hypertext transport protocol). In Fig. 8, one logical port in the logical model is converted to a port address for each instance of the module. Logical port 422 converts into physical address ports 804(1)-804(J) and logical port 440 converts into physical ports 806(1)-806(K).

The paragraph on page 28, spanning lines 6-11, has been amended as shown below:

The port table 904 tracks instances of the port in the logical model, such as ports 422 and 440 in Fig. 8. A record from this table includes such information as the port ID, the model component identity, a node ID, the network address represented by the port, the instance ID of the corresponding instance, the protocol used by the port, and an ID of the wire to which the port is connected. Again, more or fewer ~~less~~ fields may be used in other implementations.

The paragraph spanning page 30, line 20, through page 31, line 8, has been amended as shown below:

At block 1010, the management policy 702 calls the loader 722 to install software onto the allocated node and configure the node to perform the functions represented by the logical module from which the instance is created. In response, the loader 722 initializes the node by communicating with the node loader 732 ~~730~~ of the new node via the network 708 to install the appropriate software; such an operation might install an image of an operating system (e.g., Windows® NT server operating system from Microsoft Corporation). The loader 722 then loads the appropriate software and configures the node to perform the functions represented by the logical model component from which the instance is derived. For example, for an instance of an SQL module in the logical model 400, the node loader 732 loads SQL server software. The loader 722 registers the physical port addresses with the service running state 710, which records the new instance in the instance-tracking database 522.

The paragraph on page 31, spanning lines 15-18, has been amended as shown below:

At block 1014, the management policy 702 is notified by the core ~~runtime~~ logical-to-physical converter 704 that a new instance is up and running. The new instance should relieve the event or operating condition to bring operation back into compliance with the policy.

The Abstract, appearing on page 45, has been amended as shown below:

ABSTRACT

A system facilitates the design and implementation of large-scale applications, such as Internet Services and Websites, for distributed computer systems, such as server data centers, Internet data centers (IDCs), Web farms, and the like. The system has a modeling system and a deployment system. The modeling system permits developers to architect the hardware and software used to implement the applications in an abstract manner. The modeling system defines a set of components used to describe the functionality of an application. The model components are arranged and interconnected to form a scale-independent logical model of the application. Once a logical model is created, the deployment system uses the logical model to automatically deploy various computer/software resources to implement the application. The deployment system converts each of the model components into one or more instances that correspond to physical resources. ~~As one example, the resources correspond to the computer nodes of a distributed computer system that are loaded with specific types of software to implement the function represented by the model components. The deployment system initially installs the application on the computer nodes according to the logical model. It then dynamically and automatically modifies the allocation of computer nodes used to implement the application in an ongoing basis as the operating parameters of the application change.~~